A Survey of Cryptographically Difficult Problems and Quantum Algorithms

Simon Weydert
Iowa State University
Ames, IA
weyderts@iastate.edu

Abstract

This paper introduces core cryptographic problems at the heart of several influential cryptographic protocols, including the RSA and Diffie-Hellman key exchange protocols. The paper details how order finding and period measurement, a field that quantum computers excel, can be used to reduce the computational complexity of solving the prime factorization and the discrete log problem. Finally, this paper gives a brief overview of a few quantum algorithms that have been developed to solve these problems.

1 Difficult Cryptographic Problems

When discussing cryptographic protocols, their security is often derived from a single difficult problem that is computationally infeasible. In this paper we discuss two problems: The prime factorization problem and the discrete log problem. Both problems can be reduced to the order finding problem.

1.1 Order Finding and Period Finding

Many cryptographic applications derive their security from the difficulty of: factoring large composite numbers into prime factors; and finding a solution to the discrete log problem. Both of these problems can be reduced to the problem of order finding. These reductions will be discussed after a discussion on the order finding problem.

In this paper, there are two types of nomenclature used to denote modular arithmetic. Notation (1) indicates that the numbers a and b are equivalent under the modulus N. Notation (2) indicates that applying the modulus N function to a produces b.

(1) $\quad a \equiv b \pmod{N}$

(2) $\quad a \bmod N = b$

In this paper, we use the nomenclature of cyclic groups. A cyclic group $\langle g \rangle$ is any group of numbers whose elements can all be generated by a single generator g under repeated operations $\odot$.

$$\langle g \rangle = \{1, \quad g, \quad g \odot g, \quad g \odot g \odot g, \quad ...\}$$

The cyclic groups talked about in this paper are only those where $\odot$ is an exponent operator.

$$\langle g \rangle = \{1, \quad g, \quad g^2, \quad g^3, \quad ...\} = \{g^x, x \in \mathbb{Z}\}$$

The cyclic groups in this paper are finite: the cyclic group has a period such that $g^r = 1$ for some period r. In this case, r is called the order of the cyclic group. One way for this to occur is when the cyclic group operator includes a modulus, forcing the group to be finite. This can force g^x to wrap back around to the identity element of the group (the element 1) when $x = r$. In this paper, the values g and x are also assumed to be positive integers.

$$\begin{aligned}\langle g \rangle &= \{g^x \bmod N, \quad x \in \mathbb{N}, \quad g \in \mathbb{Z}^+, \quad g^r = 1\} \\ &= \{1, \quad g, \quad g^2, \quad ... \quad g^{r-1}\}\end{aligned}$$

For the rest of the paper, we produce cyclic groups by applying a modulus to a generator, where $f(x)$ is defined to produce every element of this group.

$$f(x) = g^x \bmod N$$
$$\langle g \rangle = \{f(x)\} = \{1, \quad g, \quad g^2, \quad ... \quad g^{r-1}\}$$

The order finding problem is as follows: for a cyclic group $\langle g \rangle$ with a known generator g, find the smallest r such that $g^r = 1$, where r is the period/order of $\langle g \rangle$.

Here is a short proof that the order of a cyclic group is also its period given $f(r) = g^r \bmod N = 1$:

$$\begin{aligned}f(x+r) &= g^{x+r} \bmod N \\ &= (g^x \bmod N)(g^r \bmod N) \\ &= g^x \bmod N = f(x)\end{aligned}$$

1.2 Prime Factorization Problem

The security of the RSA cryptographic protocol hinges on the difficulty of factoring a large semiprime (a composite number that is the product of two primes). With classical computers, factoring large semiprimes (and other large composite numbers) is computationally inefficient, but with an efficient quantum order-finding algorithm and classical post-processing, factoring composite numbers is made efficient [1].

This section will show that the prime factorization problem can be reduced to the order finding problem. Consider the composite positive integer N. Given r is the period of $\langle g \rangle$ and $g^r \equiv 1 \pmod{N}$ and $r/2 \in \mathbb{Z}$:

$$(g^r - 1) \bmod N = 0$$
$$(g^{r/2} - 1)(g^{r/2} + 1) \bmod N = 0$$
$$(g^{r/2} - 1)(g^{r/2} + 1) = kN$$
$$\frac{(g^{r/2} - 1)(g^{r/2} + 1)}{N} = k \in \mathbb{Z}$$

Now denote N as a product of prime factors:

$$N = p_1 p_2 p_3 \dots p_n$$
$$\frac{(g^{r/2} - 1)(g^{r/2} + 1)}{p_1 p_2 p_3 \dots p_n} \in \mathbb{Z}$$

For this division to be an integer, every prime factor of N must exist in either $(g^{r/2} - 1)$ or $(g^{r/2} + 1)$.

To find at least one of the factors shared by N with $g^{r/2} - 1$ or $g^{r/2} + 1$, perform the following operations, which can be performed quickly on classical computers:

$$\gcd(N, g^{r/2} - 1)$$
$$\gcd(N, g^{r/2} + 1)$$

It may be the case that all the prime factors exist within only $g^{r/2} - 1$ or $g^{r/2} + 1$, and thus no factors of N exist in the other. This would give us little insight; one or two of the terms could be a multiple of N, and one of them could be coprime to N. To assure that neither is a multiple of N, verify that:

$$g^{r/2} + 1 \not\equiv 0 \pmod{N}$$
$$g^{r/2} - 1 \not\equiv 0 \pmod{N}$$

The latter case does not actually need to be checked; if it were false, then r/2 would be the order of $\langle g \rangle$, not r. Thus, only the first case needs to be checked (and thus only $g^{r/2} + 1$ has the possibility to be a multiple of N). Provided the first case is not violated, then we can be confident that $\gcd(N, g^{r/2} - 1)$ and $\gcd(N, g^{r/2} + 1)$ will produce at least one non-trivial factor of N.

Picking many g such that eventually one has an order $r/2$ that is even is not difficult. Most any integer will do since $\gcd(g, N) = 1$ indicates the two are coprime and $\gcd(a, N) \neq 1$ indicates a factor of N was just found accidentally.

1.3 Discrete Log Problem

The discrete log problem is altered and extended in many cryptographic applications. The original Diffie-Hellman key exchange, Diffie-Hellman Finite Field, as well as Diffie-Hellman Elliptic Curve use variants of the discrete log problem. The former uses an exponential under a modulus, whereas the latter uses points and lines along an elliptic curve. Both of these systems form a finite cyclic group.

The general discrete log problem is as follows: given a cyclic group $\langle g \rangle$, find an unknown x given known $g, a \in \langle g \rangle$ such that $g^x = a$. Just like the prime factorization problem, assuming you can solve for the period of the cyclic group in a reasonable time, you can use the result of a period finding method to solve the general discrete logarithm problem in a computationally reasonable time [2].

The reduction of the DLP to period finding is trickier than for prime factorization. The general reduction utilizes a known order of the cyclic group, or the order found by an efficient algorithm for order finding, and is as follows.

1) Know or determine the order of the group $\langle g \rangle$.
2) Create a function $f : \mathbb{Z}_r \times \mathbb{Z}_r \to \langle g \rangle$ that encodes x in some measurable pattern, period, or frequency.
3) Ascertain that pattern using a quantum algorithm, exposing x.

An important note: $\mathbb{Z}_r = \{0, 1, \dots r-1\}$. Thus, a and b are all the unique exponents/inputs to the cyclic group; other inputs larger than r or smaller than 0 map to another input.

Additionally, some more context on cyclic groups is in order. Within a cyclic group, g^k is called a multiplicative operation (applied k times). There is also a multiplicative inverse, denoted by g^{-1}, such that $g^{-1}g = 1$ (the identity element of the group). This notation can be extended to g^{-k}, which is a multiplicative operation carried out on a multiplicative inverse: $g^{-k} = (g^{-1})^k$.

In quantum methods like Shor's algorithm, the function f is defined as $f(c_1, c_2) = g^{c_1} a^{-c_2} = g^{c_1} g^{-xc_2} = g^{c_1 - xc_2}$. It is implied here that these are group operations, which appear to operate under some of the same laws that exponents operate under. As a more concrete example, in finite field Diffie-Hellman, this function would be $f(c_1, c_2) = g^{c_1 - xc_2} \bmod p$ where the exponent is an actual exponential operator and where p is the prime modulus operating on the generator g.

2 Quantum Algorithms for Difficult Problems

The difficult problems introduced in the previous section came with some assumptions. Primarily that there was an efficient way to determine the period/order of some cyclic group or function. This is an area of computation where quantum algorithms vastly outperform classical algorithms. I will briefly touch on these: Shor's algorithms, variational quantum factoring, and Oded Regev's quantum factoring algorithm.

2.1 Shor's Algorithms

In Shor's seminal paper, Shor employs quantum Fourier sampling to extract periods from finite cyclic groups. He uses this technique to present solutions to the prime factoring problem and the discrete logarithm problem [3].

For the prime factoring problem, he presents an algorithm for extracting the order of a cyclic group that runs in polynomial time on a fault-tolerant quantum computer. As discussed above, the prime factoring problem is computationally feasible on classical computers once the order of a modular cyclic group defined by the number being factored is determined.

Shor also augments his order finding algorithm to solve the discrete log problem. His DLP solution focuses on the cyclic group generated by performing $g^r \bmod p = x$, where p is a large prime and the generator g is assumed to be a primitive of p, forcing the order of the resulting cyclic group to be $p - 1$. The goal here would be finding r given g, p, and x. This is reminiscent of the DH finite-field key exchange

referenced earlier in this paper. For other cyclic groups with unknown order, the order could be found with Shor's order finding technique.

In his DLP solution, he first generates an equal-amplitude superposition over all $a, b \in \mathbb{Z}_p$ on the vectors $|a, b, f(a, b)\rangle$ where $f(a, b) = g^a x^{-b} = g^{a-rb}$. Then he applies a quantum Fourier transform to the superposition, which exposes r after measurement and potentially several runs of this algorithm. Unlike the prime factorization algorithm, the DLP algorithm is mostly quantum with little to no classical components.

2.2 Variational Quantum Factoring

Variational quantum factoring (VQF) [4] is a quantum algorithm that was introduced to make Shor's algorithm more feasible to implement on contemporary quantum computers. A significant issue with Shor's algorithm on modern noisy intermediate-scale quantum (NISQ) computers is aggregated error: Shor's algorithm creates a very deep circuit, especially when the number being factored is large. Variational factoring was proposed as a way to realize a quantum factoring algorithm without such depth.

The first step in VQF is to represent the problem not as order finding but as an optimization problem. Specifically, as a binary optimization problem, where a composite number is phrased as the product of two binary values. The optimization condition derived from this product is encoded as a "factoring Hamiltonian." This is constructed to map the solution the system is trying to reach to the ground state of an Ising Hamiltonian.

The ground state of this Ising Hamiltonian is approached by an parameterized Ansatz state. These parameters are then brute forced to search for a state in this approximated quantum system with a close overlap with the desired ground state. This is why while Shor's algorithm has a clean Big O runtime, VQF does not.

Many of the decisions the authors of VQF made when designing the algorithm were motivated by empirical evidence and the Big O runtime of certain components of the algorithm/system. Still, the entire algorithm lacks runtime precision. For instance, there is an amount of classical preprocessing that has a quadratic Big O runtime that significantly reduces the number of qubits the system needs to produce good results. This matches the mission of their study: to create an alternative to Shor's algorithm that functions on current quantum hardware.

2.3 Oded Regev's Quantum Factoring Algorithm

Like the variational quantum factoring algorithm, Oded Regev's quantum factoring algorithm was inspired by the large gate-depth of Shor's algorithm. Unlike VQF, Regev's algorithm is designed as an algorithm with a predictable runtime and a predictable number of quantum gates, similar to Shor's. Unlike Shor's algorithm, Regev's algorithm does not rely on order finding for a cyclic group produced by a modulus. Regardless, it can be generalized to solve every hard problem mentioned in this paper.

Regev's algorithm relies on running a quantum process several times independently, then performing post-processing on the measurements of those quantum processes to find the factors of a given integer. His algorithm requires $O(n^{3/2})$ gates with roughly $O(n^{1/2})$ runs, while Shor's requires one run of a circuit demanding $O(n^2)$ gates, where n is the size of the number being factored in bits. This ideally helps with issues like decoherence and aggregated error on real hardware. This isn't to say this algorithm is a perfect improvement; Regev's requires $O(n^{3/2})$ qubits, whereas Shor's only requires $O(n)$ qubits [5].

Instead of encoding a cyclic group produced by a modulus defined by the target number N in a single qubit superposition, Regev leverages a lattice structure to produce the elements of the modular cyclic group. Then, a sublattice is defined that contains the elements of the modular cyclic group that don't satisfy the conditions for a number that could share prime factors with N (nearly identical to the case
$g^{r/2} \pm 1 \not\equiv 0 \pmod{N}$). By finding elements that belong to the larger sublattice but not the smaller sublattice, numbers that share prime factors with N can be measured and processed. At this point, the process of finding prime factors for N is essentially identical to the process Shor used; a similar set of operations to those described in section 1.2 to perform prime factorization from order finding.

While Shor's algorithm has at least been tested for small values in various labs since its inception, Regev's algorithm lacks such rigorous benchmarking. That is not to say tests don't exist; one came out a few months ago [6]. However, given the difficultly of preparing and testing these algorithms for large input sizes, the benefits and drawbacks of Regev's factoring algorithm in comparison to Shor's factoring algorithm remains to be definitively proven.

3 Concluding Remarks

The process of solving the cryptographically difficult problems utilized by asymmetric key exchange protocols with quantum computers always involves encoding some kind of period or pattern of the problem in a quantum system such that the pattern can be measured. By the nature of the finite cyclic groups used by these protocols, some kind of repeating pattern is unavoidable.

The quantum algorithms being developed to expose this latent periodicity all make tradeoffs between performance, circuit depth, versatility, and realizability. Shor's algorithms are highly versatile, but lack concrete realization in the near future. Variational quantum factorization presents an algorithm that is easier to realize on current hardware, but has no guarantees of scaling well for large inputs. Regev's factoring algorithm leverages and optimizes for lattice-based quantum computing to make strides in performance and circuit depth in theory, but its benefits remain unproven in practice.

References

[1] Quantum Computing Shor's Algorithm https://courses.cs.washington.edu/courses/cse599d/06wi/lecturenotes11.pdf

[2] The discrete log problem https://www.mccurley.org/papers/dlog.pdf

[3] Shor's algorithms https://arxiv.org/abs/quant-ph/9508027

[4] Variational quantum factoring (a newer quantum factoring algorithm) https://arxiv.org/abs/1808.08927

[5] Oded Regev's quantum factoring algorithm (a newer quantum factoring algorithm) https://arxiv.org/abs/2308.06572

[6] Implementation and Analysis of Regev's Quantum Factorization Algorithm https://arxiv.org/pdf/2502.09772